



Laurea Magistrale in informatica-Università di Salerno
Corso di *Gestione dei Progetti Software*- Prof.ssa F. Ferrucci



EVIM

ENGLISH VALIDATION
&
INTERNSHIP MANAGEMENT

Quality Plan

EVIM - English Validation & Internship Management

Riferimento	
Versione	1.2
Data	28/11/2019
Destinatario	Top Management
Presentato da	Edoardo Carpentiero – Attilio Della Greca
Approvato da	Edoardo Carpentiero – Attilio Della Greca



Revision History

Data	Versione	Descrizione	Autori
26/11/2019	1.0	Prima stesura	Edoardo Carpentiero Attilio Della Greca
27/11/2019	1.1	Revisione Standard, pratiche, convenzioni e metriche	Edoardo Carpentiero Attilio Della Greca
28/11/2019	1.2	Strumenti e tecnologie	Edoardo Carpentiero Attilio Della Greca



Sommario

Revision History.....	2
1. Introduzione	4
1.1 Scopo del documento.....	4
1.2 Evoluzione del documento.....	4
1.3 Panoramica del progetto.....	4
1.4 Riferimenti	5
1.5 Definizioni, acronimi e abbreviazioni.....	5
2. Struttura gestionale.....	6
2.1 Organigramma	6
2.2 Task	6
2.3 Ruoli e responsabilità.....	7
3. Documentazione	8
3.1 Documenti di prodotto	8
3.2 Documenti di management.....	8
4. Standard, pratiche, convenzioni e metriche.....	9
4.1 Definizione della qualità	9
4.2 Standard per la documentazione	14
4.3 Standard per gli artefatti	18
4.4 Standard di codifica	30
4.5 Standard e pratiche per il testing	30
4.6 Metriche per la valutazione del progetto	30
4.7 Metriche per la valutazione della documentazione.....	30
5. Revisioni del software	31
6. Test	31
7. Rapporto sui problemi e azioni correttive.....	32
8. Strumenti, tecniche e metodologie	32
9. Controllo dei dati multimediali.....	33
10. Controllo della fornitura.....	33
11. Collezione, manutenzione e conservazione dei dati	33
12. Training	33
13. Gestione dei rischi.....	34

1. Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è quello di definire i livelli accettabili di qualità che dovranno essere rispettati durante tutte le fasi del progetto, in modo da poter garantire la possibilità di creare artefatti che siano conformi a tali standard.

In particolare, il documento ha come obiettivi:

1. Definire le linee guida e gli standard che dovranno essere usati nella stesura della documentazione del progetto
2. Definire i criteri di qualità da applicare al progetto
3. Definire le metriche e le tecniche utilizzate per la valutazione della qualità degli artefatti dei membri del team durante l'intero progetto

1.2 Evoluzione del documento

Durante lo svolgimento del progetto vi saranno continue revisioni e modifiche del documento, volte a garantire una alta e sempre maggiore qualità degli artefatti.

1.3 Panoramica del progetto

Il Dipartimento di Informatica dell'Università degli Studi di Salerno intende semplificare ed automatizzare gli attuali processi burocratici che consentono di gestire l'attivazione, lo svolgimento e il riconoscimento dei CFU dei tirocini curriculari per i propri studenti. Tali tirocini possono essere esterni o interni svolti rispettivamente in Aziende/Enti convenzionati con il Dipartimento e in laboratori dell'Università.

Oltre alla gestione dei tirocini curriculari, il riconoscimento delle Attività Lavorative svolte dallo Studente in Aziende non convenzionate con il Dipartimento, rientra nelle attività da semplificare ed automatizzare.

La semplificazione e l'automatizzazione di tali processi è motivata dalla poca efficienza dell'iter burocratico utilizzato attualmente, a causa di continui documenti da firmare e da far firmare alle parti interessate.

Per questo motivo è stato proposto un progetto, chiamato English Validation & Internship Management, nel quale si intende sviluppare e integrare il modulo che gestisce gli aspetti inerenti alla gestione dei tirocini, chiamato Internship Management, in un sistema già presente, chiamato English Validation. Lo scopo di questa integrazione è quello di fornire al Presidente del Consiglio Didattico, all'Ufficio Carriera e allo Studente un'unica piattaforma per gestire sia gli aspetti burocratici per la gestione del tirocinio e sia al riconoscimento dei crediti per quanto riguarda la Lingua Inglese.

1.4 Riferimenti

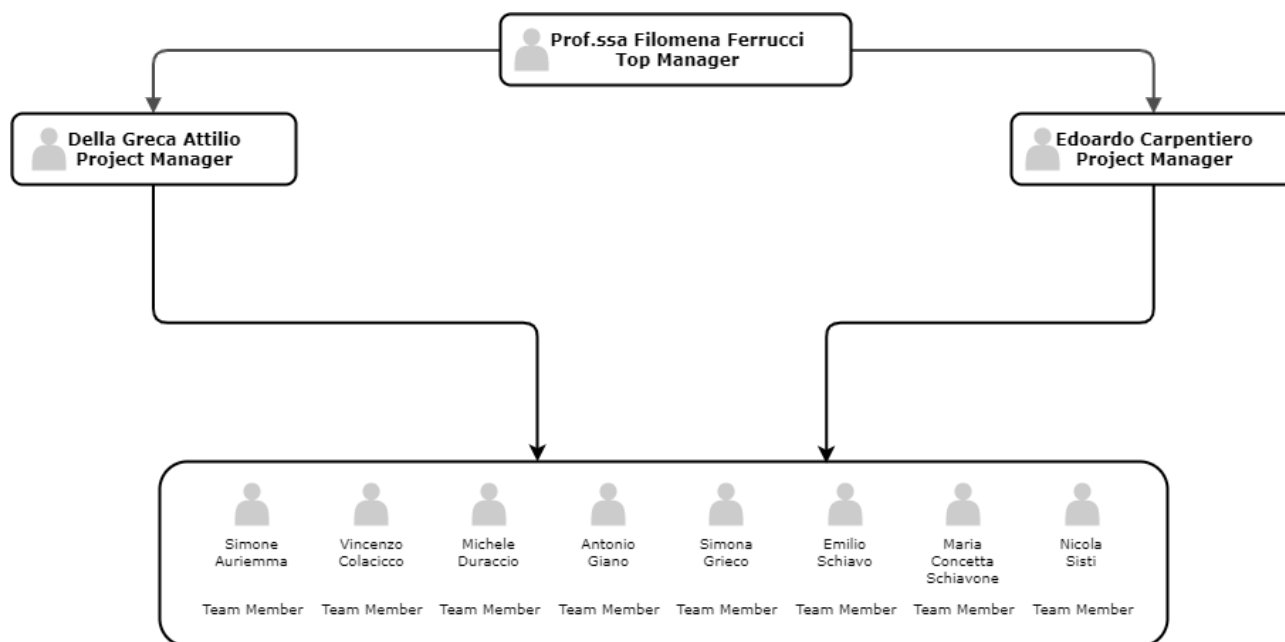
- Kathy Schwalbe, “Information Technology Project Management”, International Edition 7E, Cengage Learning, 2014;
- Bernd Bruegge, Allen H. Dutoit, “Object-Oriented Software Engineering Using UML, Patterns and Java”, Third Ed., Pearson, 2010;
- Sommerville, “Software Engineering”, Addison Wesley;
- PMBOK® Guide and Software Extension to the PMBOK® Guide, Fifth Ed., Project Management Institute, 2013
- Documentazione di Progetto

1.5 Definizioni, acronimi e abbreviazioni

- **EVIM:** English Validation & Internship Management
- **RAD:** Abbreviazione utilizzata per indicare il Requirement Analysis Document;
- **SDD:** Abbreviazione utilizzata per indicare il System Design Document;
- **TP:** Abbreviazione utilizzata per indicare il Test Plan;
- **TC:** Abbreviazione utilizzata per indicare i Test Case;
- **ODD:** Abbreviazione utilizzata per indicare l’Object Design Document
- **CP:** Abbreviazione utilizzata per indicare il Category Partition
- **IT:** Abbreviazione utilizzata per indicare gli Integration Test;
- **ITP:** Abbreviazione utilizzata per indicare l’Integration Test Plan;
- **UTR:** Abbreviazione utilizzata per indicare gli Unit Test Report;
- **TM:** Abbreviazione utilizzata per indicare i Team Member;
- **PM:** Abbreviazione utilizzata per indicare i Project Manager;
- **WBS:** Abbreviazione utilizzata per indicare la Work Breakdown Structure;
- **WBSD:** Abbreviazione utilizzata per indicare la Work Breakdown Structure Dictionary;

2. Struttura gestionale

2.1 Organigramma



2.2 Task

2.2.1 Definizione Quality Plan

La definizione del Quality Plan si basa sul SOW del progetto e le informazioni condivise con il cliente Top manager. Vengono definiti gli standard di qualità del progetto in base agli standard dell'organizzazione dettati dal Top manager, ai costi, benefici e criteri di accettazione del cliente.

2.2.2 Checklist di revisione

Le checklist di revisione viene fornita dal Top manager ai PM la quale è associata ad ogni documento prodotto dai TM. La checklist compilata da quest'ultimi consente di verificare in modo dettagliato gli artefatti riportati in ogni documento, e quindi la revisione del documento prodotto.

2.2.3 Elaborazione artefatto

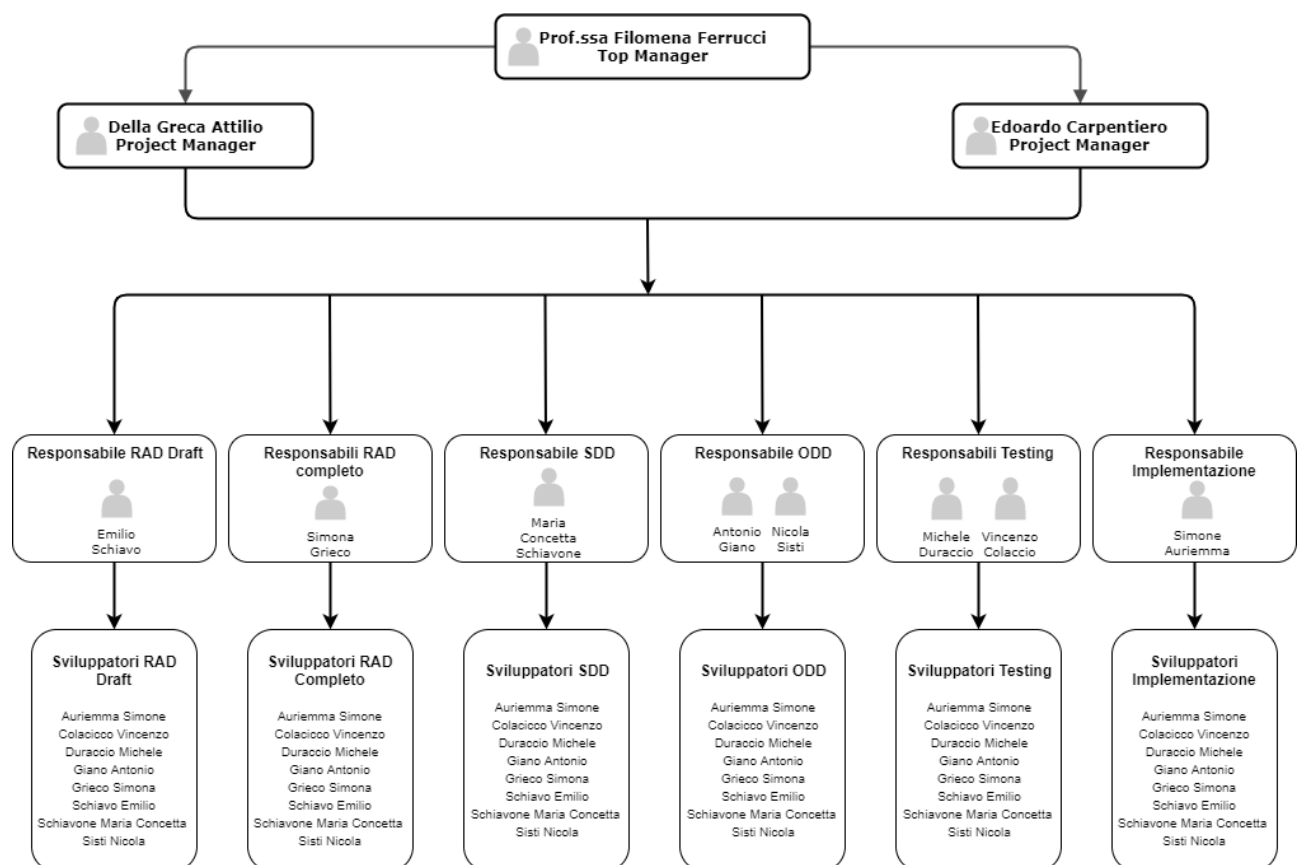
Gli artefatti prodotti dai TM dovranno essere conforme agli standard definiti nel Quality Plan.

2.2.4 Revisione e correzione artefatto

Due giorni prima della consegna il responsabile del documento dovrà procedere alla revisione e (eventualmente) dovrà rendere conforme il documento agli standard di qualità. Il giorno prima della consegna del documento i Project Manager revisioneranno il documento corretto dal responsabile, nel caso il documento presenta ancora difetti i manager li segnaleranno al responsabile che provvederà alla correzione.

2.3 Ruoli e responsabilità

Per ogni deliverable sono stati individuati dei responsabili, il cui ruolo è quello di revisionare il documento prodotto e compilare la check list. In questo modo si è cercato di responsabilizzare i TM. Quest'ultimo sarà stimolato a controllare la qualità del documento sin dalle prime fasi in modo da non avere un carico di lavoro eccessivo durante il periodo di revisione, considerando che anch'esso si occupa della stesura del documento. I ruoli e i responsabili individuati sono presenti nell'organigramma.



3. Documentazione

3.1 Documenti di prodotto

I documenti dei TM sono:

- **RAD modello funzionale:** documento usato per la raccolta e l'analisi dei requisiti funzionali e non funzionali individuati e la loro analisi sotto forma di scenari, casi d'uso e mock-up
- **RAD:** documento contenente gli artefatti del RAD modello funzionale con l'aggiunta dei modelli dinamici e ad oggetti
- **SDD:** documento che riporta la progettazione del sistema nella prima fase di modellazione, contiene una suddivisione ad alto livello del sistema
- **ODD:** documento che riporta e analizza tutte le componenti a basso livello del sistema riportandole così come saranno implementate;
- **TCS:** documento che riporta i test case;
- **CP:** documento che riporta il category partition del sistema
- **ITP:** documento che riporta la pianificazione del testing di integrazione
- **UTP:** documento che riporta la pianificazione del testing di unità
- **UTR:** documento che riporta il report test di unità.
- **TIR:** documento che riporta il report degli errori riscontrati durante la fase di testing
- **TER:** documento che riporta il report dell'esecuzione dei test case
- **TSR:** documento che riporta il report del testing

3.2 Documenti di management

I documenti prodotti dai PM sono:

- **Agenda:** documento che sintetizza tutto ciò che verrà trattato al prossimo meeting
- **Business case:** documento che riporta l'analisi finanziaria eseguita sul progetto
- **Final Project Report:** report finale del progetto
- **Post Mortem:** questionario valutativo da somministrare a fine progetto
- **Project Charter:** documento che specifica le responsabilità di ogni team member
- **SOW:** documento che specifica lo statement del progetto
- **SA:** diagramma di Gann di tutte le attività pianificate che mostrano in chiaro le date stabilite di inizio e fine attività, le tempistiche necessarie e le risorse necessarie per portare a termine tali attività
- **WBS:** strumento utilizzato per la consentire una rappresentazione gerarchica delle attività pianificate del progetto
- **WBSD:** Documento che descrive ciascun task e sottotask presente nella WBS
- **Quality Plan:** documento che specifica la gestione della qualità del progetto
- **Registro degli stakeholder:** documento che specifica tutte le persone che saranno coinvolte in maniera diretta o indiretta nel progetto
- **Risk Management Plan:** documento che specifica i rischi del progetto e la loro gestione
- **Scheduling Plan:** documento che specifica la pianificazione delle attività
- **SPMP:** documento che specifica la gestione del progetto

- **Team Contract:** documento che specifica il contratto del team
- **Test Plan:** documento che riporta la pianificazione del test e dei test case

4. Standard, pratiche, convenzioni e metriche

4.1 Definizione della qualità

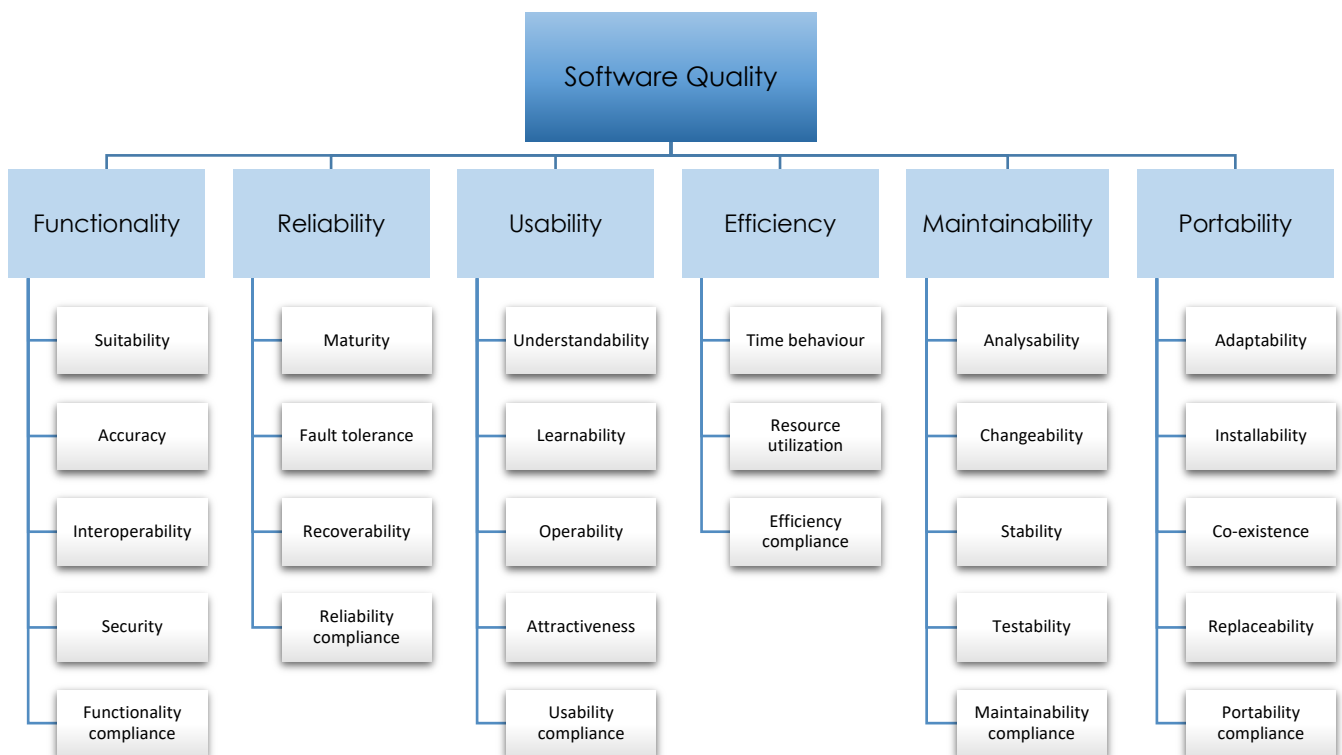
Lo standard di riferimento per la definizione del presente Quality Plan è l'ISO/IEC 9126.

Le norme ISO/IEC 9126 descrivono un modello di qualità del software, definiscono le caratteristiche che la determinano e propongono metriche per la misurazione.

Le norme relative alla qualità del software emesse da questo standard si dividono in 4 parti:

- 1 Modello della qualità del software;
- 2 Metriche esterne per la qualità;
- 3 Metriche interne per la qualità;
- 4 Metriche per la qualità in uso.

Il modello prevede che per ciascuna di esse siano associate sottocaratteristiche, dette anche attributi. Nel grafico sottostante saranno mostrate le caratteristiche e gli attributi del software proposti dal modello.



Caratteristica	Attributi	Peso	Razionale	Trade-Off
Functionality	Suitability	5	Non può essere escluso in quanto la mancanza di questo attributo rappresenterebbe un prodotto software incompleto	Budget
	Accuracy	5	Non può essere escluso in quanto la mancanza di questo attributo rappresenterebbe un prodotto software incompleto	Budget
	Interoperability	1	Il sistema dovrebbe interfacciarsi con "ESSE3" ma sappiamo già che non è possibile quindi inutile investire sull'interoperabilità	Non è possibile ottenere l'interoperabilità con il sistema "ESSE3"
Reliability	Maturity	3	Non è possibile evitare sempre gli errori	Convieni investire di più sulla tolleranza agli errori in modo da mantenere alte le prestazioni del sistema
	Fault tolerance	5	Il sistema deve garantire un'alta tolleranza ai guasti	Si preferisce investire le risorse sulla tolleranza agli errori piuttosto che sulla prevenzione agli errori
Usability	Understandability	5	Il sistema deve essere di facile utilizzo e comprensione in modo da permettere l'utilizzo anche senza effettuare un training	Budget
	Learnability	4	Il sistema deve essere di facile utilizzo senza costringere l'utente a dover memorizzare molte informazioni di utilizzo	Soddisfazione del cliente e degli utenti finali
	Operability	4		Soddisfazione del cliente e degli utenti finali

	Attractiveness	3	Si vuole fornire un'interfaccia grafica che punti più alla semplicità di utilizzo piuttosto che alle componenti grafiche	Budget
Efficiency	Time behaviour	2	Il team non si concentra sulle prestazioni, però ovviamente si devono avere dei tempi di risposta accettabili	Il team si concentrerà sulla manutenibilità
	Resource utilization	2	Le risorse e lo schedule sono già state allocate	
Maintainability	Analysability	1	Non sono stati previsti strumenti di diagnostica	Budget
	Changeability	5	Il team si impegna a rendere manutenibile il sistema e tutti gli artefatti correlati	Budget
	Stability	4	Il team ha puntato sulla manutenzione, quindi eventuali modifiche dovranno mantenere la stabilità	Soddisfazione del cliente e degli utenti finali
	Testability	2	Nel progetto è già previsto il testing standard	Risorse già allocate
Portability	Adaptability	1	Essendo una piattaforma web l'adattabilità del sistema è già prevista	Feature già presenti
	Installability	2	Essendo una piattaforma web è prevista solo una piccola fase di configurazione iniziale	Feature già presenti
	Co-existence	1	Essendo una piattaforma web, la coesistenza con altre applicazioni indipendenti è già prevista	Feature già presenti
	Replaceability	1	Non esiste un sistema simile da dover sostituire	Feature già presenti
	Portability compliance	1	È già prevista la portabilità del sistema	Feature già presenti

4.1.1 Functionality

La funzionalità rappresenta la capacità del software di fornire funzioni, espresse ed implicite, necessarie per operare in determinate condizioni, cioè in un determinato contesto. Gli attributi del software richiesti da questa caratteristica sono:

1. **Suitability:** rappresenta la capacità di un prodotto software di fornire un appropriato insieme di funzioni che permettano agli utenti di svolgere determinati task e di raggiungere gli obiettivi prefissati;
2. **Accuracy:** rappresenta la capacità di un prodotto software di fornire i risultati o gli effetti attesi con il livello di precisione richiesta;
3. **Interoperability:** rappresenta la capacità di un prodotto software di interagire con uno o più sistemi specificati;
4. **Functionality compliance:** rappresenta la capacità di un prodotto software di aderire a standard, convenzioni e regolamenti di carattere legale o prescrizioni simili che abbiano attinenza con la funzionalità

4.1.2 Reliability

L'affidabilità rappresenta la capacità di un prodotto software di mantenere il livello di prestazione quando viene utilizzato in condizioni specificate. Possibili limitazioni all'affidabilità del software possono essere causate da errori nei requisiti, nella progettazione e nel codice. Le evidenze di tali errori possono essere rilevate a seconda delle condizioni in cui il prodotto è utilizzato oppure alle opzioni scelte, piuttosto che al momento in cui è utilizzato.

Gli attributi richiesti da tale caratteristica sono:

1. **Maturity:** rappresenta la capacità di un prodotto software di evitare che si verifichino errori o siano prodotti risultati non corretti in fase di esecuzione;
2. **Fault tolerance:** rappresenta la capacità di un prodotto software di mantenere il livello di prestazioni in caso di errori nel software o di violazione delle interfacce specificate;

4.1.3 Usability

L'usabilità rappresenta la capacità di un prodotto software di essere comprensibile, di poter essere studiato, di risultare attraente da parte di un utente sotto determinate condizioni.

Alcuni aspetti della “funzionalità”, “affidabilità” ed “efficacia” possono influire sull'usabilità del prodotto software.

Gli attributi della caratteristica usabilità sono:

- **Understandability:** rappresenta la capacità di un prodotto software di permettere all'utente di capire le sue funzionalità e come poterla utilizzare con successo per svolgere particolari task in determinate condizioni di utilizzo. Essa dipende dalla documentazione disponibile e dall'impressione iniziale che si riceve dal prodotto;

- **Learnability:** rappresenta la capacità di un prodotto software di permettere all'utente di imparare ad utilizzare l'applicazione. L'attributo corrisponde alla funzionalità relativa all'apprendimento del software;
- **Operability:** rappresenta la capacità di un prodotto software di permettere all'utente di utilizzarlo e di controllarlo. Gli aspetti relativi alla funzionalità, modificabilità, adattabilità ed installabilità del software possono influire sull'operabilità del prodotto. L'operabilità del software fa riferimento anche alle aspettative dell'utente sulla sua controllabilità, tolleranza ai guasti e conformità;
- **Attractiveness:** rappresenta la capacità di un prodotto software di risultare "attraente" per l'utente. La qualità è relativa alla progettazione dell'aspetto grafico delle sue interfacce, all'utilizzo dei colori e delle immagini, ecc.;

4.1.4 Efficiency

L'efficienza rappresenta la capacità di un prodotto software di realizzare le funzioni richieste nel minor tempo possibile ed utilizzando nel miglior modo le risorse necessarie, quando opera in determinate condizioni. Le risorse includono altri prodotti software e la configurazione hardware e software del sistema.

Gli attributi relativi all'efficienza del software sono:

- **Time behaviour:** rappresenta la capacità di un sistema software di fornire appropriati tempi di risposta, tempi di elaborazione e quantità del lavoro eseguendo le funzionalità previste sotto determinate condizioni di utilizzo.
- **Resource utilization:** rappresenta la capacità di un prodotto software di utilizzare un appropriato numero e tipo di risorse quando esegue le funzionalità previste sotto determinate condizioni di utilizzo. Le persone sono incluse nella definizione di risorse;

4.1.5 Maintainability

La manutenibilità rappresenta la capacità di un prodotto software di essere modificato. Le modifiche possono includere correzioni o adattamenti del software a modifiche negli ambienti, nei requisiti e nelle specifiche funzionali.

Gli attributi previsti per la manutenibilità sono:

- **Analysability:** rappresenta la capacità di un prodotto software di poter effettuare la diagnosi sul software ed individuare le cause e di errori o malfunzionamenti;
- **Changeability:** rappresenta la capacità di un prodotto software di consentire lo sviluppo di modifiche al software originale. L'implementazione include modifiche al codice, alla progettazione ed alla documentazione. Nel caso in cui le modifiche debbano essere fatte dagli utenti, la modificabilità può influire sull'operabilità del prodotto;

- **Stability:** rappresenta la capacità di un prodotto software di evitare effetti non desiderati in seguito di modifiche al software;
- **Testability:** rappresenta la capacità di un prodotto software di consentire la verifica e la validazione del software modificato, cioè di eseguire i test;

4.1.6 Portability

La portabilità rappresenta la capacità di un prodotto software di poter essere trasportato da un ambiente ad un altro. L'ambiente include aspetti organizzativi e tecnologici.

Gli attributi previsti per la portabilità sono:

- **Adaptability:** rappresenta la capacità di un prodotto software di essere adattato a differenti ambienti senza richiedere azioni specifiche diverse da quelle previste dal software per tali attività. L'adattabilità include la scalabilità delle capacità interne del prodotto (campi delle schermate, tabelle, volumi delle transazioni, formato dei report, ecc.);
- **Installability:** rappresenta la capacità di un prodotto software di essere installato in un determinato ambiente;
- **Co-existence:** rappresenta la capacità di un prodotto software di coesistere con altre applicazioni indipendenti in ambienti comuni e di condividere le risorse;
- **Replaceability:** rappresenta la capacità di un prodotto software di sostituire un altro software specifico indipendente, per lo stesso scopo e nello stesso ambiente; ad esempio sostituire un prodotto software con una nuova versione;
- **Portability compliance:** rappresenta la capacità di un prodotto software di aderire a standard e convenzioni relative alla portabilità;

4.2 Standard per la documentazione

4.2.1 Standard per il processo di documentazione

Il processo per la realizzazione della documentazione relativa al progetto deve seguire questo iter:

1. Creare una versione iniziale del documento;
2. Effettuare una revisione della versione;
3. Aggiungere eventuali cambiamenti individuati nella fase di revisione;
4. Scrivere una nuova versione del documento;
5. Ripetere i passi da 2 a 4 fin quando il documento non viene approvato;
6. Realizzare la versione finale del documento;
7. Apportare le correzioni al layout;
8. Sottomettere il documento sulla piattaforma e-learning;

4.2.2 Standard per i documenti

4.2.2.1 Insieme di stili di base per i documenti

La stesura dei documenti sarà guidata da template forniti dal PM. In ogni template saranno definiti la struttura e gli stili che i membri del team devono seguire per redigere il documento.

Tutta la documentazione dovrà essere scritta in lingua italiana (esclusi diagrammi che richiedono la lingua inglese a causa del software già esistente) utilizzando il software di word processing Microsoft Word.

Ogni documento deve contenere:

- Un frontespizio in cui sia presente:
 - Il logo del progetto;
 - Il titolo del documento (acronimo e nome per esteso);
 - La data dell'ultima modifica;
 - La versione del documento;
 - Destinatario;
 - Chi lo ha presentato;
 - Chi l'ha approvato.
- Una Revision history in cui si presente:
 - Data della stesura del documento;
 - La versione del documento;
 - La descrizione di eventuali cambiamenti introdotti;
 - Il nome dell'autore del documento;
- Un indice dei contenuti.

Ogni pagina deve essere dotata di:

- Un' intestazione in cui sia presente:
 - Il logo del dipartimento di informatica dell'Università di Salerno;
 - La seguente intestazione:

Laurea Magistrale in informatica- Università di Salerno
Corso di *Gestione dei Progetti Software* – Prof.ssa F. Ferrucci
- Un piè di pagina in cui sia presente:
 - Il logo del progetto sulla sinistra;
 - Il titolo del documento immediatamente alla destra del logo;
 - Il numero di pagina sulla destra.

Il nome del file deve contenere SiglaProgetto_SiglaDocumento_Vers.x.yz

Il formato dei caratteri dei documenti dovrà seguire le seguenti convenzioni:

	Font	Grandezza	Grassetto	Corsivo	Sottolineato	Colore	Allineamento
Titolo Documento	Century Gothic	48	No	No	No	Blu	Destra
Sottotitolo Documento	Garamond	20	No	No	No	Blu	Destra
Titolo Capitoli	Century Gothic	18	No	No	Si	Blu	Sinistra
Titolo Paragrafi	Garamond	13	Si	No	No	Nero	Sinistra
Sottotitoli Paragrafi	Garamond	12	Si	Si	No	Nero	Sinistra
Testo	Garamond	12	No	Se necessario	No	Nero	Giustificato
Intestazione	Garamond	12	No	No	No	Nero	Allinea al centro
Piè di pagina	Century Gothic	8	No	No	No	Blu	Destra
Intestazione Tabelle	Century Gothic	12	Si	No	No	Bianco	Allinea al centro
Contenuto Tabelle	Century Gothic	11	No	No	No	Nero	Allinea al centro
Sommario	Century Gothic	11	No	No	No	Nero	Sinistra

4.2.2.2 Dettagli Colori

Di seguito vengono descritti i dettagli dei colori utilizzati:

- Blu, colore: 1, 50% più scuro;
- Nero, colore: automatico;
- Bianco: colore: sfondo 1.

4.2.2.3 Stile tabella

Di seguito vengono riportate le caratteristiche degli stili utilizzati per le tabelle:

- Tabella: griglia 5 scura – colore 1;
- Sfondo intestazione: Blu, colore: 1, 25% più scuro;
- Testo intestazione: Bianco: colore: sfondo 1.

4.2.2.4 Frontespizio

Ogni documento deve presentare un frontespizio standard che presenta le seguenti caratteristiche:

- Il logo del progetto in alto e centrato;
- Il titolo del documento (acronimo e nome per esteso) immediatamente sotto al logo;
- Una tabella riassuntiva in cui siano presenti:
 - La data dell'ultima modifica;
 - La versione del documento;
 - Destinatario;
 - Chi lo ha presentato;
 - Chi l'ha approvato

4.2.2.5 Regole sugli identificatori dei documenti

L'assegnamento dei nomi ai documenti prodotti durante le fasi di sviluppo del software è molto importante per la tracciabilità dei documenti stessi; pertanto la sintassi di base da seguire per identificare i documenti è la seguente:

<Sigla progetto>_<acronimoDocumento>_V_<versioneDocumento>.

È importante, inoltre, mantenere la tracciabilità anche per gli artefatti; per gli artefatti il modello è:

Nome artefatto: <acronimo.Artefatto>_<numero.Artefatto>.

Gli acronimi definiti per gli artefatti sono:

- RF: Requisito Funzionale;
- RNF: Requisito non Funzionale;
- SC: Scenario;
- UC: Use Case;
- UCD: Use Case Diagram;
- CD: Class Diagram;
- SD: Sequence Diagram;
- NP: Navigation Path;
- UI: Mock-up.

4.2.2.6 Regole per riflettere i cambiamenti tra le versioni di un documento

Ogni qualvolta viene apportata una modifica sostanziale ad un documento deve essere aggiornata la **Revision History** ad esso associato, specificando la data della modifica, la versione del documento, la descrizione della modifica e gli autori.

In questo modo sarà possibile mantenere una versione distinta del documento per ogni entry della revision history; ogni versione del documento avrà la forma x.y, dove x rappresenta la versione

del documento e y la sotto versione che viene a crearsi quando la modifica al documento non è particolarmente ampia.

In questo modo, avendo a disposizione una copia per ogni versione del documento, possiamo sfruttare lo strumento di confronto, messo a disposizione da Microsoft Word, per poter visivamente costatare ciò che differenzia le due versioni.

4.2.3 Standard per lo scambio di documenti

I documenti devono essere redatti utilizzando il programma Microsoft Word 2016, in modo da non avere nessun problema di portabilità. Tutti i documenti vengono condivisi tra i membri del team e il PM utilizzando Google Drive. La sottomissione dei documenti, invece, avviene mediante la piattaforma e-learning di informatica.

4.3 Standard per gli artefatti

4.3.1 Convenzione per i requisiti funzionali

I requisiti funzionali avranno come acronimo RF;

La descrizione dei requisiti funzionali deve seguire il seguente template:

RF_<acronimoGestione>: <nomeGestione>

<Descrizione della gestione>

Elenco tabulato RF_< acronimoGestione >_<numerazione>: <nomeFunzionalità>

<Descrizione funzionalità>

La descrizione della funzionalità deve essere della forma:

<Il sistema dovrà><descrizione funzionalità>

4.3.2 Convenzione per i requisiti non funzionali

I requisiti non funzionali avranno come acronimo RNF seguito da un numero identificativo progressivo.

L'identificazione dei requisiti non funzionali dovrà fare riferimento al modello FURPS+, ma solamente per le categorie ritenute di interesse per lo sviluppo del progetto.

4.3.3 Convenzione per gli scenari

Ogni scenario deve far riferimento ad uno e un solo requisito.

Il nome dello scenario deve essere costituito da:

SC_<acronimoGestione>_X: <nome dello scenario>, con x numero del requisito a cui fa riferimento.

Le regole per la realizzazione degli scenari sono di seguito riassunte:

- La tabella per ogni scenario deve essere composta da: nome scenario, partecipanti, flusso degli eventi.
- I partecipanti devono essere preceduti dal rispettivo nome proprio.
- Il flusso degli eventi deve iniziare con l'interazione di un partecipante.
- Il flusso degli eventi deve essere strutturato in modo da mostrare le operazioni dei partecipanti allineate a sinistra.
- Il flusso degli eventi deve essere strutturato in modo da mostrare le operazioni del sistema allineate a destra.
(se si hanno problemi di resa grafica, il flusso può essere anche descritto in successione, precisando sempre l'attore che effettua l'operazione).
- Nel flusso degli eventi le operazioni dell'utente devono iniziare con il nome proprio di un partecipante.
- Nel flusso degli eventi le operazioni del sistema devono iniziare con "Il sistema..." o con il nome del sistema stesso.

4.3.4 Convenzione per gli use case

Ogni use case deve far riferimento ad uno e un solo requisito.

Il nome dello use case deve rispettare questo modello:

UC_<acronimoGestione>_X: <nome dello use case>, con X numero del requisito funzionale a cui fa riferimento.

Le regole per la realizzazione degli use case sono di seguito riassunte:

- La tabella per ogni caso d'uso deve essere composta secondo il template allegato.
- Il nome del caso d'uso deve includere un verbo e deve essere univoco.
- Il nome del caso d'uso deve indicare cosa intende fare l'attore.
- Il nome dell'attore deve essere un sostantivo, che indica un ruolo rispetto all'uso del sistema.
- I nomi degli attori e dei casi d'uso devono basarsi su elementi del dominio dell'applicazione, anche i termini del flusso di eventi deve far riferimento al dominio del problema.
- Il flusso degli eventi deve iniziare con l'interazione dell'attore (triggering event)
- Un caso d'uso deve descrivere una transizione utente completa.
- Un caso d'uso non deve descrivere un'interfaccia del sistema, meglio descrivere con mock-up.
- Fare riferimento ai mock-up scrivendo nel punto in cui si fa riferimento all'interfaccia grafica "cfr. (MU_x)" dove x assume i valori 1,2,3...



- La descrizione di un caso d'uso non deve superare le 2 pagine, altrimenti deve essere decomposto in casi d'uso più piccoli.
- Nel caso d'uso base deve essere invocato il caso d'uso incluso in un punto specifico.
- L'evento che determina l'attivazione del caso d'uso che estende deve essere indicato nella condizione di ingresso del caso d'uso che estende.
- Non bisogna usare la forma passiva (le relazioni casuali tra le varie parti del flusso degli eventi devono essere chiari).
- Descrivere tutti i flussi di eventi (non solo quello principale).
- Definire i termini importanti nel glossario.

Nella pagina successiva è mostrato il template per gli use case.

Identificativo UC_Pack_TUC	<i>Nome del caso d'uso</i>		<i>Data</i>	<i>dd/mm/aa</i>
			<i>Vers.</i>	<i>0.00.000</i>
			<i>Autore</i>	<i>Cognome Nome</i>
Descrizione	<i>Descrizione generale del caso d'uso (scope).</i>			
Attore Principale	Nome Interessi nell'esecuzione del caso d'uso			
Attori secondari	Nome Interessi nell'esecuzione del caso d'uso			
Entry Condition	Descrizione.			
Exit condition On success	Descrizione.			
Exit condition On failure	Descrizione.			
Rilevanza/User Priority	Priorità attribuita al caso d'uso dagli utenti.			
Frequenza stimata	N/giorno			
Extension point	<condition, UCE> Indica che questo use case è esteso dallo use case UCE quando "condition" è true. ¹			
Generalization of	UCG Indica che UCG è padre di questo use case			
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO				
1	Attore:	STEP 1.		
2	Sistema:	STEP 2		
3	Attore:	STEP 3		
4	Sistema:	<i>include (UCIn)</i>		
...				
I Scenario/Flusso di eventi Alternativo: primo scenario alternativo				
1.1	Sistema:	Elenco delle azioni da eseguire come alternativa a quanto prescritto nel primo passo.		
II Scenario/Flusso di eventi Alternativo: Descrizione				
3.1	Sistema:	Elenco delle azioni da eseguire come alternativa a quanto prescritto nel II passo.		
...				
I Scenario/Flusso di eventi di ERRORE: Descrizione				
2.1	Sistema:	Elenco delle azioni da eseguire nel caso in cui si verifichi una condizione di errore durante l'esecuzione del secondo passo.		
II Scenario/Flusso di eventi di ERRORE: Descrizione				
4.1	Sistema:	Elenco delle azioni da eseguire nel caso in cui si verifichi una condizione di errore durante l'esecuzione del quarto passo.		
...				
Note				
5	Annotazioni relative al punto 5 dello scenario principale.			
Special Requirements				

¹ Specificare nello use case esteso i casi d'uso estendenti, da un punto di vista formale, costituirebbe un problema: lo use case esteso non ha conoscenza di quanti e quali casi d'uso lo estendono... Però, in questo caso, si tratta esclusivamente di una convenzione che agevola produzione e manutenzione della documentazione.


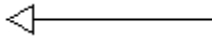
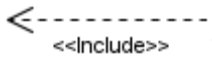
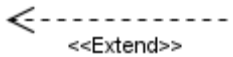
4.3.5 Convenzione per gli use case diagram

Gli use case diagram devono far riferimento ad un solo package di requisiti.

Il nome degli use case diagram deve rispettare questo formato:

UCD_<acronimoGestione>: <nome use case diagram>.

Le regole da rispettare per la stesura degli use case diagram sono le seguenti:

- Il diagramma dei casi d'uso deve essere composto da attori (omini stilizzati) e casi d'uso (ovali). Gli attori devono essere collegati con una linea continua ai casi d'uso a cui partecipano. Il confine del sistema deve essere indicato tracciando un rettangolo (o package) che racchiude tutti i casi d'uso.
- Fornire i diagrammi dei casi d'uso a diversi livelli di astrazione.
- L'attore principale dovrebbe essere posizionato al lato sinistro del rettangolo.
- Sotto l'attore deve essere indicato il nome dell'attore.
- Gli attori secondari devono essere posizionati al lato destro del rettangolo.
- Prima del nome dell'attore che indica un sistema deve essere aggiunto lo stereotipo **<<system>>**.
- L'attore tempo deve essere denominato **Time**.
- Applicare **<<include>>** quando si conosce esattamente quando invocare il caso d'uso.
- Applicare **<<extend>>** quando un caso d'uso può essere invocato in varie parti del flusso di eventi del caso d'uso.
- La **generalizzazione degli attori** deve essere indicata con una freccia del tipo:
 . Il verso della freccia deve andare dall'attore specializzato verso l'attore generico.
- La **generalizzazione dei casi d'uso** deve essere indicata con una freccia del tipo:
 . Il verso della freccia deve andare dal caso d'uso specializzato verso il caso d'uso generico.
- L'**inclusione** dei casi d'uso deve essere indicata con una freccia del tipo:
 . Il verso della freccia deve andare dal caso d'uso incorporante verso il caso d'uso incluso.
- L'**estensione** dei casi d'uso deve essere indicata con una freccia del tipo:
 . Il verso della freccia deve andare dal caso d'uso che estende verso il caso d'uso esteso.

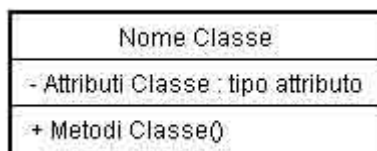
4.3.6 Convenzione per i class diagram

Il nome del Class Diagram deve rispettare questo modello:

CD_<acronimoGestione>: <nome del class diagram>.

Con questo diagramma si vanno a specificare ad alto livello le classi che faranno parte del sistema e le loro interazioni.

Una classe viene rappresentata come segue:

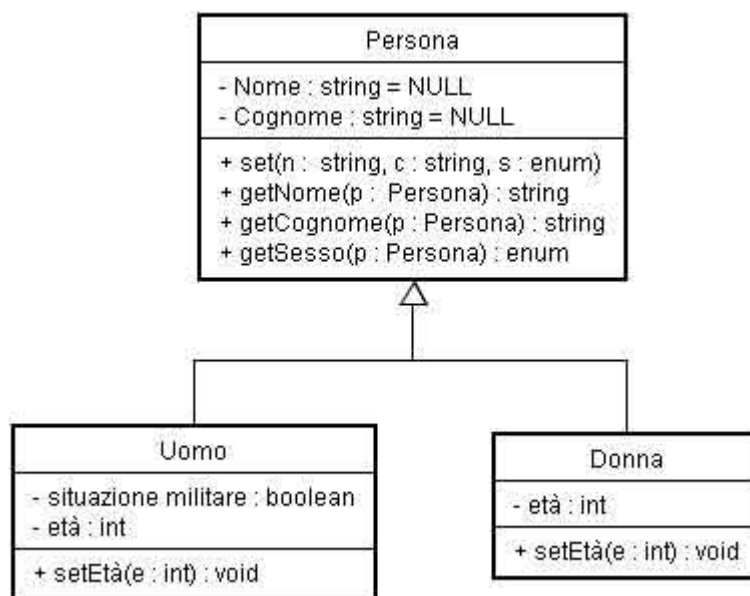


La visibilità di ogni attributo o metodo è specificata in UML attraverso l'uso di vari simboli:

- **“-” visibilità privata:** l'attributo è accessibile solo dall'interno della classe usando i propri metodi.
- **“+” visibilità pubblica:** l'attributo o il metodo è accessibile anche dall'esterno.
- **“#” visibilità protetta:** l'attributo o il metodo viene ereditato da tutte le classi da questa derivate.

Relazioni tra classi ed oggetti

- **Generalizzazione-specializzazione:** con questa relazione è possibile creare una superclasse da cui far derivare in seguito classi figlie. Evitando di specializzare da subito tutti gli attributi di una classe, si ha il vantaggio di poter inquadrare sin dall'inizio il domino di applicazione. Questa relazione si traduce nei linguaggi di programmazione orientati agli oggetti con l'uso dell'ereditarietà.

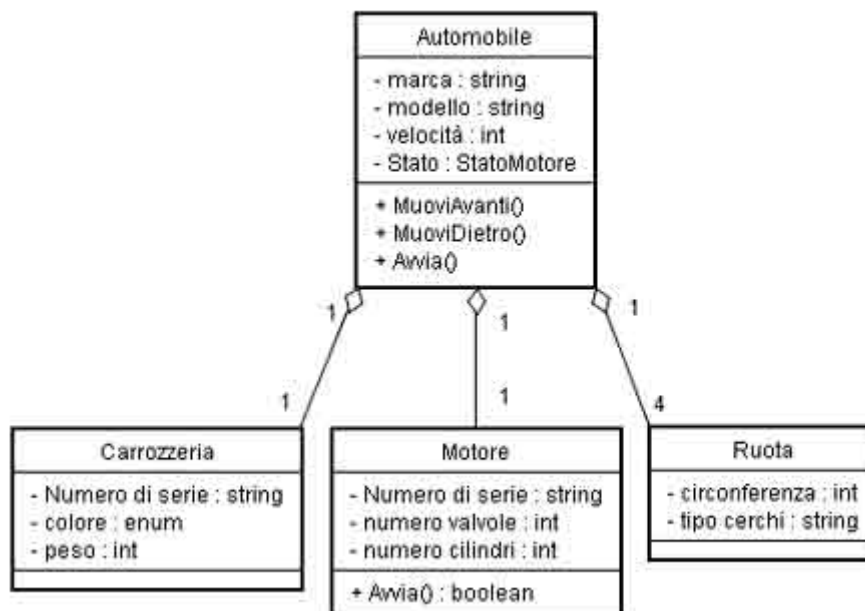


- **Aggregazione:** con questa relazione si possono creare aggregati di oggetti; il vantaggio di usare questa relazione consiste nel fatto che, in presenza di un oggetto composto da più parti, non andiamo a gestire l'oggetto nella sua totalità ma in funzione delle sue parti; in questo modo si va a gestire piccole classi piuttosto che una sola grande classe.

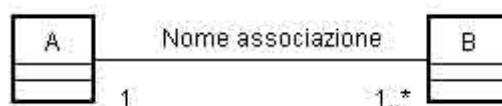
Esistono due tipi di aggregazione:

- **Lasca** indica che l'oggetto "contenuto" ha vita propria anche senza l'oggetto "contenitore"; essa viene rappresentata con un rombo vuoto rivolto verso la classe contenitore;
- **Stretta** invece indica che l'oggetto non ha vita propria, quindi deve essere distrutto assieme al "contenitore"; essa viene rappresentata con un rombo pieno rivolto verso la classe contenitore;

Un'ulteriore aggiunta a questa relazione è data dalla **molteplicità**. La molteplicità, posta in prossimità della freccia che rappresenta l'aggregazione, indica il numero di occorrenze necessitiamo di una classe per ogni aggregazione; se si vogliono indicare molte occorrenze si utilizza il simbolo *****.



- **Associazione:** con questa associazione si esprime i legami che ci sono fra le classi. È caratterizzata da:
 - **un nome**, rappresenta il nome che attribuiamo all'associazione;
 - **una molteplicità**, ha lo stesso significato dell'aggregazione;
 - **una direzione**, indica il verso di lettura dell'associazione, se abbiamo una linea senza frecce indica la bidirezionalità;



4.3.7 Convenzione per i sequence

Ogni sequence diagram deve far riferimento ad uno e un solo requisito.

Il nome del sequence diagram deve rispettare questo modello:

SQ_<acronimoGestione>_X: <nome del sequence diagram>, con X numero del requisito funzionale a cui fa riferimento.

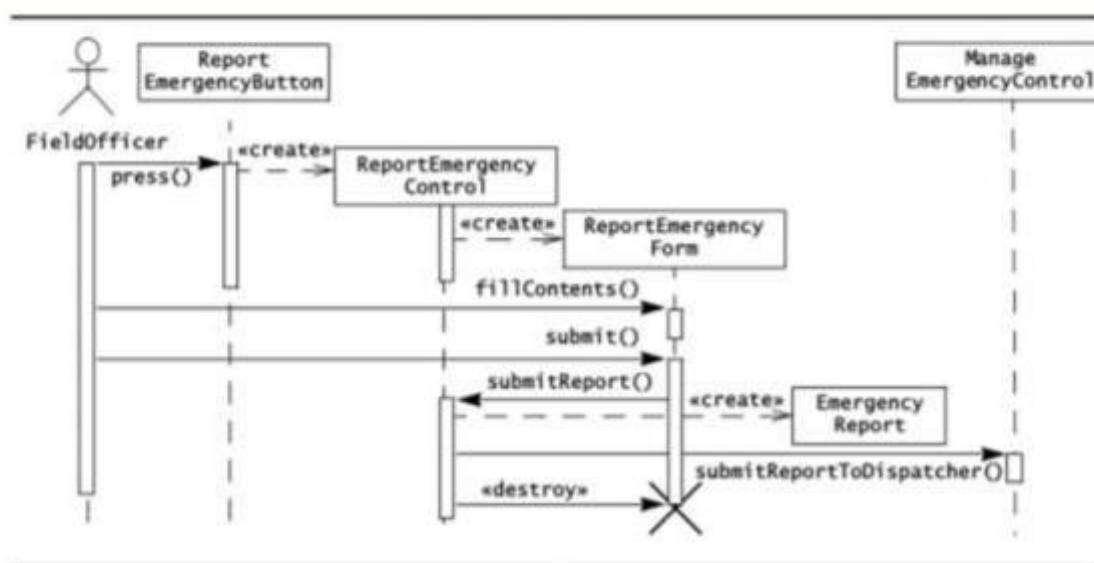


Figure 5-8 Sequence diagram for the ReportEmergency use case.

Layout

- Le colonne rappresentano gli oggetti che partecipano al caso d'uso:
- La prima colonna deve corrispondere all'attore che ha avviato il caso d'uso.
- La seconda colonna deve corrispondere ad un oggetto boundary con cui l'attore interagisce per iniziare il caso d'uso.
- La terza colonna dovrebbe corrispondere all'oggetto control che gestisce il caso d'uso.

Creazione

- Gli oggetti control vengono creati all'inizio del sequence diagram e si estendono per tutta la durata dello stesso.
- Gli oggetti control possono creare altri oggetti boundary e possono interagire con altri oggetti control.

- In cima al diagramma (vedi figura 5-8, a destra dell'attore) si trovano gli oggetti che esistono prima che il flusso abbia inizio (nell'esempio l'attore, FieldOfficer, clicca sul bottone ReportEmergencyButton; il bottone esiste dapprima che l'attore invii il primo messaggio).

Accesso

- Gli oggetti Entity sono accessibili da oggetti Control e Boundary.
- Gli oggetti Entity non possono chiamare oggetti Control o oggetti Boundary.

Istanze di classi

- Le istanze vengono rappresentate da rettangoli con il nome della classe e l'identificatore dell'oggetto sottolineato, oppure, semplicemente con un nome dal quale si evince che si sta considerando un'istanza della classe (es. ReportEmergencyButton).

Attori

- Il sequence diagram deve essere composto da attori (es. l'omino stilizzato dell'esempio 5-8). È necessario indicare il nome dell'attore.
- L'attore tempo deve essere denominato Time.
- Gli attori vengono posizionati sulla sinistra, con le frecce di interazione verso gli oggetti del sistema.
- Gli attori possono anche non essere riportati nel caso in cui il caso d'uso venga avviato da un altro caso d'uso.

Messaggi

- Le frecce orizzontali tra le colonne rappresentano i messaggi inviati da un oggetto a un altro. I messaggi sono rappresentati come frecce da un attore a un oggetto, o tra due oggetti.

pressButton2() →

- I messaggi di return sono rappresentati con frecce tratteggiate.

← GMTTime —

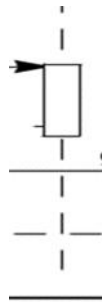
- I messaggi sincroni terminano con una freccia triangolare piena.
- I messaggi asincroni terminano con una freccia semplice.
- Una freccia circolare (che parte e arriva nella stessa box di attivazione) rappresenta un messaggio che ha per destinatario lo stesso mittente. Rappresentato nel seguente modo:



- L'ordine dei messaggi (dall'alto verso il basso) ricalca l'ordine sequenziale con il quale vengono scambiati.
- Gli oggetti creati durante l'interazione sono preceduti da un messaggio di <<create>>.
- Gli oggetti distrutti durante l'interazione sono evidenziati con una croce e preceduti da un messaggio <<destroy>>

Lifelines (linee di vita)

Le lifelines sono linee tratteggiate verticali che partono dal rettangolo rappresentativo dell'oggetto e giungono fino in fondo al diagramma. Indicano il periodo temporale di vita dell'oggetto, della creazione alla distruzione.



- La linea di vita di un oggetto definito staticamente parte dalla cima del diagramma.
- Costruzione di un nuovo oggetto non presente nel sistema fino a quel momento: l'oggetto viene collocato nell'asse temporale in corrispondenza dell'invocazione del costruttore tramite il messaggio <<create>> (o new)
- La linea tratteggiata indica il tempo nel quale l'oggetto può ricevere un messaggio.
- Distruzione di un oggetto presente nel sistema fino a quel momento: si rappresenta con una X posta in corrispondenza della life-line dell'oggetto. Da quel momento in poi non "è legale" invocare alcun (comportamento) dell'oggetto distrutto.

4.3.8 Convenzione per gli statechart diagram

Il nome dello Statechart Diagram deve rispettare questo modello:

SCD_<acronimoGestione>: <nome dell'entità coinvolta>.

Con questi diagramma si descrive un comportamento dinamico del sistema, modellando dinamicamente ciò che accade in un determinato oggetto. In sostanza si rappresenta il comportamento di un determinato oggetto di una classe in relazione anche ad input esterni.

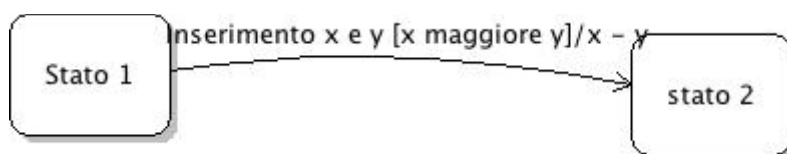
Questi diagrammi sono caratterizzati da quattro componenti principali:

- **Stato iniziale:** è lo stato iniziale dell'oggetto; è rappresentato con un cerchio colorato di nero;
- **Stato generico:** rappresenta un generico stato assunto dall'oggetto. È rappresentato con un rettangolo con gli angoli stondati;
- **Transazioni:** collegano uno stato sorgente ad uno stato destinazione nel quale l'oggetto può andare. Sono rappresentate con delle frecce con all'estremità una freccia che punta verso lo stato destinazione;
- **Stato finale:** è lo stato finale dell'oggetto; è rappresentato dal simbolo dello stato iniziale inscritto in un cerchio più grande a sfondo bianco.

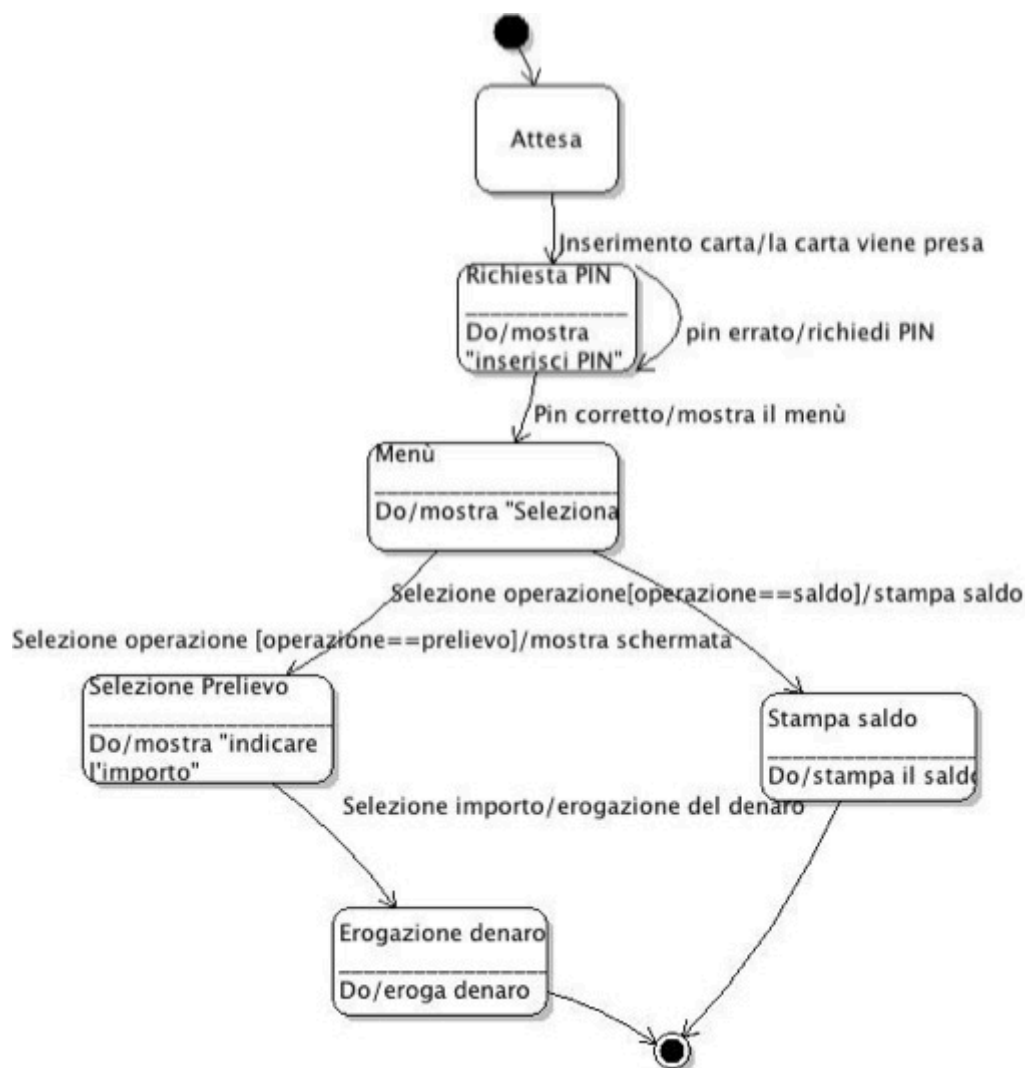
Per quanto riguarda le transazioni si segue la seguente sintassi:

Evento [guardia]/azione 1; azione 2;...;azione n

- **Evento:** nome dell'evento preso in considerazione;
- **Guardia:** è un costrutto opzionale e sostanzialmente controlla che determinate condizioni siano verificate prima di effettuare una transazione;
- **azione 1; azione 2;...;azione n:** rappresentano tutte le azioni che l'oggetto compie dopo la transazione.



Di seguito è mostrato un esempio di statechart diagram.



4.3.9 Convenzione per i mockup

Ogni mock-up deve essere specifico a uno scenario per poter facilitare la lettura del flusso degli eventi dello scenario di riferimento.

Il nome del mock-up deve rispettare questo modello:

UI_<acronimoGestione>_X: <nome del mock-up>, con X numero del requisito funzionale a cui fa riferimento.



4.4 Standard di codifica

Per quanto riguarda la codifica, lo standard utilizzato si baserà su quello fornito dall'organizzazione W3C.

4.5 Standard e pratiche per il testing

Si rimanda alla lettura del documento **EVIM_TP_Vers.1.1.pdf**.

4.6 Metriche per la valutazione del progetto

Per la valutazione del progetto bisogna avere branch Coverage dei casi di test: almeno 75% ed una buona manutenibilità.

4.7 Metriche per la valutazione della documentazione

Ogni documento che sarà rilasciato sarà sottoposto ad una doppia fase di verifica e validazione. Innanzitutto, è il responsabile del documento che controllerà la qualità del documento stesso e ne fornirà una copia al PM; questa prima fase sarà guidata dalle linee guida e checklist che il PM fornirà al responsabile del documento.

Una volta consegnato il documento al PM, quest'ultimo darà via ad una nuova fase di verifica durante la quale controllerà che le checklist siano realmente verificate.

Un documento supererà questo controllo di qualità se almeno il **75%** dei parametri contenuti nelle checklist saranno soddisfatti.

5. Revisioni del software

Di seguito viene mostrato il piano per la revisione degli artefatti e dei documenti.

Artefatto	Data di revisione	Metodo di revisione	Azioni ulteriori
DRAFT RAD	13 Novembre 2019	Lettura	Check e commenti da parte dei PM e revisione successiva, con modifiche
RAD	23 Novembre 2019	Check-list	Check e commenti da parte dei PM e revisione successiva, con modifiche
SDD	5 Dicembre 2019	Check-list	Check e commenti da parte dei PM e revisione successiva, con modifiche
TP & TCS	5 Dicembre 2019	Lettura	Check e commenti da parte dei PM e revisione successiva, con modifiche
ODD	15 Dicembre 2019	Check-list	Check e commenti da parte dei PM e revisione successiva, con modifiche
ITC	15 Dicembre 2019	Lettura	Check e commenti da parte dei PM e revisione successiva, con modifiche
Codice sorgente	8 Gennaio 2020	Testing Automatico e Manuale; Prova del sito	Check e commenti da parte dei PM e revisione successiva, con modifiche

6. Test

Si rimanda alla lettura del documento **EVIM_TP_Vers.1.1.pdf**.

7. Rapporto sui problemi e azioni correttive

Il progetto verrà supervisionato in modo costante lungo le diverse fasi di progetto. I PM si rendono disponibili per mitigare qualsiasi dubbio in merito alle attività da svolgere tramite colloqui formali e informali in base alle esigenze.

Per quanto concerne la documentazione, durante il meeting di introduzione ai nuovi documenti da produrre vengono forniti i giusti template e le linee guida da seguire lungo la redazione dello stesso. Si assume che i TM abbiano le competenze necessarie per produrre contenuti proficui e ben formulati. A seguito del meeting, i PM guidano da vicino i TM fornendo metodologie e punti di riflessione per la stesura dei contenuti. Il primo controllo quindi avviene durante la prima stesura del documento stesso. Ogni task assegnato viene notificato sull'apposito spazio dedicato su Trello. Ogni utente allocato per un determinato task può notificare il completamento. Infine, raggiunto il termine ultimo per la produzione del documento, viene assegnata la revisione complessiva del documento ai TM. Coloro che ne verranno incaricati saranno forniti di una Checklist che permette di focalizzarsi sui punti critici da curare all'interno del documento stesso.

Per quanto concerne il codice di sviluppo, oltre ai tool definiti per il controllo dello stile e del codice, verranno sfruttate le funzionalità disposte dal CVS Github per il versioning del sistema. Ogni volta che viene effettuata una Pull Request, i PM si impegnano alla revisione del codice e al Merge del progetto qualora la richiesta venga accettata e validata.

8. Strumenti, tecniche e metodologie

Il controllo della qualità e della produttività è un processo che avviene ogni settimana, per quel che riguarda i TM, con la compilazione da parte dei PM di un documento di valutazione per ogni singolo membro del team.

Oltre a questo, i PM ogni settimana compileranno una valutazione generale dell'andamento del progetto nella sua interezza. Per quanto riguarda il controllo della qualità rispetto ai deliverables, si procederà, ogni qual volta un artefatto sarà pronto, ad effettuarne un controllo preliminare da parte dei TM (e in caso di necessità o delucidazioni, da parte dei PM).

Una volta che tutti gli artefatti che compongono un documento saranno pronti e il documento verrà strutturato e messo insieme, si procederà con il controllo da parte del reviewer (scelto a rotazione tra i membri del team) che, con il supporto delle checklist, procederà alla correzione del documento. Il reviewer potrà interrogare i TM che si sono occupati di un particolare artefatto in caso di delucidazioni e/o correzioni.

Una volta compiuta questa operazione il documento revisionato verrà inserito nella cartella dei documenti revisionati in GoogleDrive dove i PM procederanno alla correzione. A questo punto i PM commenteranno il documento utilizzando le funzionalità offerte da Word Online. I TM, osservando i commenti lasciati dai PM, effettueranno le dovute correzioni necessarie. Questa operazione verrà iterata finché il documento non soddisferà i requisiti di qualità necessari. A quel punto i PM daranno la loro accettazione e invieranno il documento al Top Manager tramite la piattaforma di e-learning.

9. Controllo dei dati multimediali

Tutti i dati multimediali utili allo sviluppo del progetto, siano essi di supporto per il team o necessari per il prodotto finale, verranno mantenuti in servizi di cloud Storage e Backup (Google Drive) in modo tale che essi possano essere facilmente reperibili e modificabili da tutti i membri del team, anche se si trovino in mobilità.

I PM potranno anch'essi accedere a tali dati in modo tale da poterli visionare, modificare ed in generale monitorare.

10. Controllo della fornitura

La qualità per quanto riguarda la realizzazione e interazione con il sistema è dovuta a Bootstrap, un framework HTML che permette di strutturare in maniera molto rapida una pagina HTML, esso mette a disposizione anche molti template se gli sviluppatori si attengono quanto più è possibile agli standard del template essi realizzeranno facilmente una applicazione usabile e responsive.

11. Collezione, manutenzione e conservazione dei dati

Come strumento per la raccolta e conservazione dei dati viene usata una cartella condivisa su GoogleDrive. La cartella condivisa è organizzata secondo un'infrastruttura di cartelle che raggruppano informazioni relative ai documenti, alle minute, alle valutazioni, materiale utile, materiale relativo all'implementazione. All'interno di ogni cartella vengono conservate anche le diverse versioni dello stesso documento. Inoltre, è disponibile sul servizio di hosting per progetti GitHub un repository che contiene tutti i deliverables del progetto, documentazione e codice inclusi.

12. Training

Poiché le tecnologie impiegate nello sviluppo del progetto rientrano nelle skill dei TM, nel progetto non sono state istanziate ore di training. L'unica attività di training pianificata riguarda l'utilizzo del tool **Enterprise Architect**, per la rappresentazione grafica dei modelli di sistema sviluppati.

13. Gestione dei rischi

Si rimanda alla lettura dei documenti:

- **EVIM_RMP_Vers.0.2.pdf**
- **EVIM_RR_Vers.1.0.xlsx**

14. Glossario

- **Artefatto:** Output ottenuto alla fine di un task;
- **Check:** Sinonimo di Controllo;
- **Codice sorgente:** Testo di un algoritmo di un programma scritto in un linguaggio di programmazione da parte di un programmatore in fase di programmazione, compreso all'interno di un file sorgente. Esso definisce dunque il flusso di esecuzione del programma stesso;
- **Framework:** piattaforma che funge da strato intermedio tra un sistema operativo e il software che lo utilizza;
- **Skill:** capacità o abilità nel fare una determinata cosa
- **Linee guida:** Insieme di raccomandazioni sviluppate sistematicamente, sulla base di conoscenze continuamente aggiornate e valide, redatto allo scopo di rendere appropriato, e con un elevato standard di qualità, un comportamento desiderato
- **Linguaggio di programmazione:** linguaggio formale dotato di una sintassi ben definita che viene utilizzato per scrivere programmi che realizzano algoritmi
- **Qualità:** Nozione alla quale sono ricondotti gli aspetti del progetto suscettibili di classificazione o di giudizio
- **Test:** Esperimento variamente espletato allo scopo di saggiare, mediante determinate reazioni, l'entità o la consistenza di un'attitudine o di una capacità individuale;
- **Tool:** piccolo programma di ausilio per attività specifiche, in genere fornito a corredo di pacchetti software
- **Training:** Attività volte all'insegnamento o al potenziamento di determinate conoscenze utili per lo svolgimento del progetto
- **Reviewer:** Membro del team che si occupa della revisione di un documento o artefatto
- **Revisione:** Esame o controllo, per lo più periodico, inteso a verificare il grado dell'efficienza, della funzionalità, della corrispondenza a determinati requisiti, in quanto può implicare apporto di modifiche o di correzioni
- **Task:** Attività assegnata ad uno dei TM;
- **Difetto:** Mancata compiutezza, sufficienza o efficienza di qualcosa;



15. Procedure di aggiornamento del piano

Questo documento sarà aggiornato ogni qualvolta sarà necessario. Tutti gli aggiornamenti apportati al documento saranno documentati nella Revision History, situata all'inizio del documento.